APPENDIX

A RELATED WORKS

FastSecAgg (Kadhe et al., 2020) replaces the quadratic-cost Shamir secret sharing of SECAGG with an FFT-based multi-secret scheme, lowering per-client complexity to $O(N \log N)$ at the expense of reduced dropout tolerance and weaker privacy guarantees.

LightSecAgg (So et al., 2022) cuts overhead by directly reconstructing the aggregate mask of the surviving clients via Lagrange Coded Computing (Yu et al., 2019), preserving the privacy and dropout-resilience of earlier schemes while still incurring multi-phase interaction costs.

Flamingo (Ma et al., 2023) amortizes the setup across training iterations and introduces a small decryptor committee to help the server strip aggregated masks, thereby reducing repeated communication in long-lived deployments. Collectively, these variants mitigate—but do not fully eliminate—the synchronization and recovery burdens that arise in large, dynamic client populations.

Willow (Bell-Clark et al., 2024), a type of one-shot protocol, adopts a static, stateful committee that must execute two setup rounds followed by two decryption rounds (threshold decryption and key-share recovery for dropped members). It also introduces auxiliary verifier parties to audit server behavior. As a result, Willow's approach incurs additional communication phases and a separate verifier committee.

TACITA (Mugunthan et al., 2025) moves away from committee-centric cryptography and incorporates a single-server asynchronous SA protocol that performs the entire summation over encrypted client updates. Privacy and correctness against a malicious server are enforced with threshold homomorphic encryption, zero-knowledge proofs, and verifiable computation. This introduces substantial overhead due to ciphertext expansion, intensive key management, and costly proof generation.

HierarchicalSA (Zhang et al., 2025) employs a three-layer topology (clients $\rightarrow$ relays $\rightarrow$ server), where clients mask their updates with correlated random keys (without secret sharing) and relays aggregate their assigned subsets before forwarding partial sums. The server then applies a linear decoding transform to recover the global sum. Although the construction achieves information-theoretic privacy, it does not specify dropout handling or adversarial robustness and thus is not a practically deployable SA protocol.

B THEORETICAL ANALYSIS

In this section, we provide the theoretical guarantees of the DISAGG protocol, starting with privacy and cryptographic preliminaries followed by analyzing convergence in FL.

B.1 Differential Privacy

A randomized mechanism $\mathcal{M} : \mathcal{X}^n \rightarrow \mathcal{Y}$ is (ε, δ)-differentially private if for all adjacent datasets $D, D' \in \mathcal{X}^n$ differing in one record and all measurable $S \subseteq \mathcal{Y}$,

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta. \tag{1}$$

In *Central DP (CDP)*, the server aggregates raw client updates and outputs a perturbed sum $S_t + Z$, where $S_t = \sum_{i \in C_t} x_i$ is the iteration-t sum over participating clients C_t and Z is noise calibrated to the sensitivity (e.g., Gaussian) (Abadi et al., 2016). In *Local DP (LDP)*, each client privatizes x_i with its own (ε, δ) mechanism before transmission, typically requiring more noise and reducing utility (Wei et al., 2019). *Distributed DP (DDP)* combines secure aggregation with distributed noise addition: each client adds an independent noise share to its update so that, when summed via SA, the aggregate attains the target CDP noise level (e.g., by splitting the Gaussian noise across clients), thereby removing the need for a trusted server to add noise centrally (Kasiviswanathan et al., 2011; Geyer et al., 2018).

B.2 Secret Sharing

We introduce threshold secret sharing because DISAGG splits each client's update across A Aggregators so that (i) up to t_c colluding Aggregators learn nothing about any individual update, and (ii) any t_r Aggregators can enable reconstruction of the needed sum. Over a prime field $\mathbb{F}_p$, we use the (t_c, t_r, A) Shamir scheme (Shamir, 1979):

- SHARE$(s; t_c, t_r, A)$: sample a random polynomial $f(X) \in \mathbb{F}_p[X]$ of degree $t_r - 1$ with $f(0) = s$, and output the shares $\{s^{(j)} := f(j)\}_{j \in [A]}$.
- COEFF(S): for $S = \{i_1, \ldots, i_{t_r}, \ldots\} \subseteq [A]$ with $|S| \geq t_r$, compute Lagrange coefficients $\lambda_{i_j} = \prod_{\zeta \in [t_r] \setminus \{j\}} \frac{i_\zeta}{i_\zeta - i_j}$ for $j = 1, \ldots, t_r$.
- RECONSTRUCT$(\{s^{(j)}\}_{j \in S})$: if $|S| \geq t_r$, return $\sum_{j \in \{i_1, \ldots, i_{t_r}\}} \lambda_{i_j} \cdot s^{(i_j)}$, yielding the encoded secret s.

This ensures correct reconstruction and information-theoretic privacy against any coalition of at most t_c malicious parties.

B.3 Convergence Analysis

To analyze convergence, we adopt the standard framework used in FedAvg for non-independent and identically distributed (non-IID) data (Li et al., 2020; Yu et al., 2018). We outline four commonly used assumptions, followed by a fifth that models partial client participation, where N out of N_T total devices are selected uniformly at random without replacement in each iteration. Since our approach preserves

the original FedAvg update rule, the theoretical analysis applies directly to our setting. From now on T denotes the total number of local stochastic gradient updates.

Assumption 1 (Smoothness): Each local objective F_k is L-smooth, i.e., for all vectors $v, w \in \mathbb{R}^d$, and for all $k \in \{1, \ldots, N_T\}$:

$$F_k(v) \leq F_k(w) + \langle \nabla F_k(w), v - w \rangle + \frac{L}{2}\|v - w\|_2^2. \quad (2)$$

Assumption 2 (Strong Convexity): Each F_k is μ-strongly convex, meaning that for all $v, w \in \mathbb{R}^d$:

$$F_k(v) \geq F_k(w) + \langle \nabla F_k(w), v - w \rangle + \frac{\mu}{2}\|v - w\|_2^2. \quad (3)$$

Assumption 3 (Bounded Gradient Variance): Let ξ_t^k be a stochastic sample drawn uniformly at random from the local dataset of client k. Then the variance of the stochastic gradients is bounded:

$$\mathbb{E}\left[\left\|\nabla F_k(w_t^k, \xi_t^k) - \nabla F_k(w_t^k)\right\|^2\right] \leq \sigma_k^2. \quad (4)$$

Assumption 4 (Bounded Gradient Norm): The expected squared norm of the stochastic gradients is uniformly bounded:

$$\mathbb{E}\left[\left\|\nabla F_k(w_t^k, \xi_t^k)\right\|^2\right] \leq G^2, \quad (5)$$

for all $k = 1, \ldots, N_T$ and $t = 1, \ldots, T-1$.

Assumption 5 (Partial Client Participation): At each iteration t, a subset $S_t \subseteq [N_T]$ of N clients is selected uniformly at random without replacement. The data distribution is assumed to be balanced, i.e., $p_k = \frac{1}{N_T}$ for all k. The server aggregates client models using:

$$w_t \leftarrow \frac{N_T}{N} \sum_{k \in S_t} p_k w_t^k. \quad (6)$$

Quantifying Non-IIDness: To capture the degree of data heterogeneity across clients, let F^* denote the global minimum of F, and F_k^* the minimum of each local objective F_k. We define the heterogeneity gap as:

$$\Gamma = F^* - \sum_{k=1}^{N_T} p_k F_k^* \quad (7)$$

When data are IID, $\Gamma \to 0$ as the number of samples increases. In contrast, a nonzero Γ indicates Non-IIDness, with its magnitude reflecting the extent of distributional divergence across clients.

Under these assumptions, the following convergence guarantee can be established for the FedAvg algorithm in our setting.

Theorem 2: Let Assumptions 1 to 4 hold, with constants L, μ, σ_k, and G defined therein. Define the condition number $\kappa = \frac{L}{\mu}$, $\xi = \max\{8\kappa, E\}$, and set the learning rate to $\eta_t = \frac{2}{\mu(\xi+t)}$. Assume Assumption 5 holds, and let $C = \frac{N_T - N}{N_T - 1} \cdot \frac{4E^2G^2}{N}$. Then:

$$\mathbb{E}[F(w_T)] - F^* \leq \frac{\kappa}{\xi + T - 1}\left(\frac{2(B+C)}{\mu} + \frac{\mu\xi}{2}\mathbb{E}\|w_1 - w^*\|^2\right) \quad (8)$$

where

$$B = \sum_{k=1}^{N_T} p_k^2 \sigma_k^2 + 6L\Gamma + 8(E-1)^2 G^2. \quad (9)$$

Here, E is the number of local iterations per one training iteration.

Remark 1 (General Convex and Non-Convex Objectives): While Theorem 1 addresses the strongly convex case, analogous convergence results can be established for general convex and non-convex objectives by leveraging the analysis framework from previous work.

Remark 2 (Client-Specific Dropout Behavior): In realistic scenarios, clients may have varying dropout probabilities, which violates the assumption of uniformly random client selection and complicates the theoretical guarantees. Nonetheless, empirical evidence suggests that FedAvg continues to converge even under such heterogeneous participation patterns.

Remark 3 (Choice of E): As shown in (Li et al., 2020), the total communication rounds T/E is a non-monotonic function of E, initially decreasing and then increasing. This suggests that overly small or large values of E may incur high communication costs, and that an optimal choice of E exists.

Remark 4 (Choice of N): The convergence rate exhibits only weak dependence on the number of active clients N. This implies that the participation ratio N/N_T can be kept small to mitigate straggler effects, without significantly impacting convergence.

Remark 5 (Error Rate): The convergence rate is $\mathcal{O}(1/T)$, meaning that the expected gap to the optimum satisfies:

$$\mathbb{E}[F(w_T)] - F^* \leq \frac{\text{const.}}{T} \quad (10)$$

Hence, increasing the total number of steps T leads to a linear decrease in worst-case error.

Next we point out the analysis of (Li et al., 2020) that diminishing learning rates are crucial for the convergence of FedAvg in the Non-IID setting.

Table 1. Extended computational and communication complexity for OPA and DISAGG. N is the number of selected clients, A is the Aggregator group size, M is the model size, ρ is the packing factor of OPA, and λ is the key size.

	Client	**Committee**	**Server**
OPA (Comp.)	$O(\lambda M + \frac{\lambda}{\rho} A \log A + A + A^2)$	$O(\lambda N)$	$O(\lambda M + NM + \frac{\lambda}{\rho} A \log A + A^2)$
OPA (Comm.)	$O(M + \frac{\lambda}{\rho} A + A)$	$O(N + N\frac{\lambda}{\rho} + \frac{\lambda}{\rho})$	$O(NA + NM + 2AN\frac{\lambda}{\rho} + A\frac{\lambda}{\rho})$
DisAgg (Comp.)	$O(\frac{M}{\rho} A log A + A^2)$	$O(N\frac{M}{\rho})$	$O(\frac{M}{\rho} A \log A + A^2)$
DisAgg (Comm.)	$O(\frac{M}{\rho} A + A)$	$O(\frac{NM}{\rho} + N + \frac{M}{\rho})$	$O(2AN\frac{M}{\rho} + NA + \frac{M}{\rho} A)$

Table 2. Altered complexity table for OPA and DISAGG, using $O(A^2)$ instead of $O(A \log A)$ for the secret sharing and reconstruction operations. This setting corresponds to the implementation complexity used in our simulations.

	Client	**Committee**	**Server**
OPA (Comp.)	$O(\lambda M + \frac{\lambda}{\rho} A^2 + A + A^2)$	$O(\lambda N)$	$O(\lambda M + NM + \frac{\lambda}{\rho} A^2 + A^2)$
OPA (Comm.)	$O(M + \frac{\lambda}{\rho} A + A)$	$O(N + N\frac{\lambda}{\rho} + \frac{\lambda}{\rho})$	$O(NA + NM + 2AN\frac{\lambda}{\rho} + A\frac{\lambda}{\rho})$
DisAgg (Comp.)	$O(\frac{M}{\rho} A^2 + A^2)$	$O(N\frac{M}{\rho})$	$O(\frac{M}{\rho} A^2 + A^2)$
DisAgg (Comm.)	$O(\frac{M}{\rho} A + A)$	$O(N\frac{M}{\rho} + N + \frac{M}{\rho})$	$O(2AN\frac{M}{\rho} + NA + \frac{M}{\rho} A)$

Theorem 3 (Effect of Learning Rate Decay): FedAvg does not necessarily converge to the optimal solution under fixed learning rates when $E > 1$. Let $\tilde{w}^*$ denote the solution obtained by FedAvg with constant step size η, and let w^* be the optimal point. Then,

$$\|\tilde{w}^* - w^*\|_2 = \Omega((E-1)\eta) \cdot \|w^*\|_2 \tag{11}$$

up to constant factors. This construction highlights the necessity of diminishing step sizes in the Non-IID setting.

Remark 6 (Learning Rate Schedule): When using a decaying learning rate and $E > 1$, FedAvg converges to the optimum. In contrast, with a fixed learning rate and $E > 1$, the algorithm does not converge to the optimum.

C COMPLEXITY ANALYSIS

C.1 Breakdown of Theoretical Complexities

Regular client: A client has a pair of private-public keys for communication with all Aggregators, and exchanges public keys with the Aggregators via the server. It sends its own with $\mathcal{O}(1)$ time complexity and receives the public key of all Aggregators with $\mathcal{O}(A)$ complexity, where $A < N$ is the size of the Aggregator group. Each client generates the shares of their model with complexity $\mathcal{O}(M \log A)$, where M is the model size (i.e., the number of model parameters). This complexity arises from the use of Fast Fourier Transform (FFT) (Cooley & Tukey, 1965) which can be used in theory for the polynomial evaluation. The total communication cost is $\mathcal{O}(M + A)$.

Aggregator: An Aggregator, similarly to the other clients, participates in the one-time key exchange: it broadcasts a single public key and (via the server) receives N client public keys to set up encrypted channels, yielding the N term. It then receives one secret share of size $\approx M/A$ from each of the N clients, so the total payload per Aggregator is $N \cdot (M/A)$ field elements, giving communication $\mathcal{O}((M/A + N)$. To compute its partial sum, the Aggregator adds these N shares entrywise (each of length M/A), which costs $\mathcal{O}(N \cdot M/A)$ arithmetic.

Server: The server transfers all public keys between N clients and A Aggregators with cost $\mathcal{O}(NA)$, the N models each of size M with $\mathcal{O}(NM)$, and the aggregated parts from the Aggregators with $\mathcal{O}(M)$. The total communication cost is $\mathcal{O}(NM + NA)$. For the reconstruction of the aggregated model, the computation complexity is the same as the client's sharing cost, in order to combine the shares: $\mathcal{O}(M \log A)$.

C.2 Refining Complexities

In this section, we provide the detailed comparison table of the extended complexities for DISAGG and OPA, as discussed in main paper (Table 1). We also provide in Table 2 a slightly modified version, where the complexity of secret sharing and reconstruction for both schemes is set to $O(A^2)$ instead of $O(A \log A)$. This alteration reflects the complexity used in our simulations for implementation simplicity. Since this change affects both protocols symmetrically, it does not influence the conclusions of the comparison.

C.3 Alternative Network Speeds

We complement the main 5G-oriented evaluation with experiments under reduced client bandwidth representative of 4G and 3G connectivity. Since DISAGG incurs additional Aggregator-facing communication, lower uplink capacity disproportionately affects its relative performance. Nevertheless, DISAGG continues to achieve a (reduced) speedup over OPA in both alternative settings.

Figures 3 and 4 report (for each network profile) (i) the

optimal number of Aggregators A selected independently for OPA and DISAGG across (M, N) pairs under the same optimization criterion as in the main text, and (ii) the corresponding relative speedup of DISAGG over OPA. The 4G configuration uses 200 kBps upload and 2 MBps download per client; the 3G configuration uses 50 kBps upload and 500 kBps download per client. The server bandwidth is held fixed at the 5G setting. As expected, reduced client bandwidth narrows the performance gap, yet DISAGG still exhibits a positive speedup in these regimes.

C.4 Aggregator Stragglers

Assuming that server bandwidth is not a bottleneck (as multiple servers can be deployed in parallel), the per-round latency is determined by the slowest surviving Aggregator. This effect can be equivalently modeled by treating Aggregators as clients with heterogeneous network speeds (e.g., 3G/4G/5G). We evaluate the expected speedups under such heterogeneous settings in Figures 3 and 4 as discussed above, and further extend our experimental evaluation on heterogeneous clients in Figure 8. Even when 40% of Aggregators use 4G connections instead of 5G, DISAGG continues to achieve a speedup over OPA. Moreover, due to its high dropout tolerance (see main paper), DISAGG can accommodate a finite round timeout to handle extreme stragglers, and is therefore not significantly impacted by Aggregator stragglers or client heterogeneity.

C.5 Minimum Aggregators Analysis

In this section, we present contour plots that illustrate the minimum number of Aggregators A (calculated for minimum packing factor $\rho = 1$) across (M, N) pairs for both DISAGG and OPA under the 5G setting. The improvement of DISAGG over OPA is shown on the right of Figure 1, where performance gains range from $0.03\times$ to $2.15\times$. To align with the configuration used in the OPA paper (Karthikeyan & Polychroniadou, 2024), we additionally report results for $\rho = 16$, corresponding to OPA's packing choice. Figure 2 presents this comparison under identical packing assumptions, with speedup factor from $0.5\times$ to $3.9\times$.

C.6 Packing Optimization

A potential optimization, left as future work, is to pack multiple 32-bit secrets into a single 128-bit field element. Since summing N client inputs requires an additional $\lceil \log_2 N \rceil$ bits of headroom to prevent overflow, one could encode two 32-bit model parameters (plus the required padding) within each 128-bit field element, effectively halving the Aggregator download burden, while maintaining correctness and security.

C.7 Byzantine Fault Tolerance

We present theoretical complexity plots under the Byzantine fault-tolerance (BFT) constraint, where the minimum committee size is chosen to satisfy $\gamma + \delta \leq 1/3$ (as discussed in main paper). For illustration, we instantiate $\gamma = \delta = 0.1$. Figure 5 reports the optimal number of Aggregators A for DISAGG and OPA across (M, N), together with the speedup of DISAGG over OPA, under the BFT constraint. As expected, enforcing BFT increases the required committee size and leads to an overall reduction in the relative performance of DISAGG compared to the non-BFT case.

D ADDITIONAL EXPERIMENTS

D.1 Training and Quantization

We conducted experiments to test the effect of quantization in our secure aggregation protocol. Figure 6 shows the training results for the models and datasets that are used in the section 5.4. In this setup we train using only plaintext FL but with two options. The first is to keep the quantization as in the secure aggregation protocols, and the second is to do the training directly with the model parameters as floats, as the nornal FL does, without security. The first is denoted with the letter 'Q' for quantization depicted on the dataset names and the latter with 'F' for floats. The plaintext field we used for quantization was 53 bits and as we can see in this Figure 6 the effect in the final accuracy for every round is negligible, with an exception for CIFAR100 dataset with TinyNet model. In this case the difference in the final accuracy is $\approx 4\%$. The main cause for this deviation is the clipping of the model parameters prior to translating floats to integers. The clipping range is set to [-2.0, 2.0] for all models.

D.2 Training Timings

We measured the timings for every phase during one round of FL for the secure aggregation protocols DISAGG and OPA along with simple aggregation PLAINTEXT. The results are presented in Tables 3 and 4, reporting the timings for training a DistilBERT model with the SST2 dataset, and EfficientNet model with the CelebA dataset respectively. DISAGG only adds up to 20% overhead while OPA can add up to 190% overhead compared to PLAINTEXT.

D.3 Grid Search for Large-scale Experiments

We present a grid search over the parameter ρ for both OPA and DISAGG to produce the results reported in main paper. The grid search highlights the importance of tuning the ρ for each method. In contrast, ρ was kept fixed in (Karthikeyan & Polychroniadou, 2024).

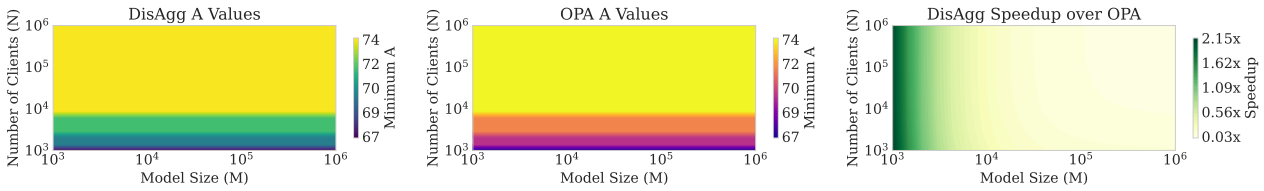


Figure 1. Contour plot showing the minimum number of Aggregators A (calculated for minimum packing factor $\rho = 1$) for DISAGG and OPA and the resulting speedup of DISAGG over OPA across (M, N) under a 5G client connectivity assumption (2 MBps upload / 20 MBps download), with $k = 0.3$ and $k_{\text{comp}} = 0.66$.

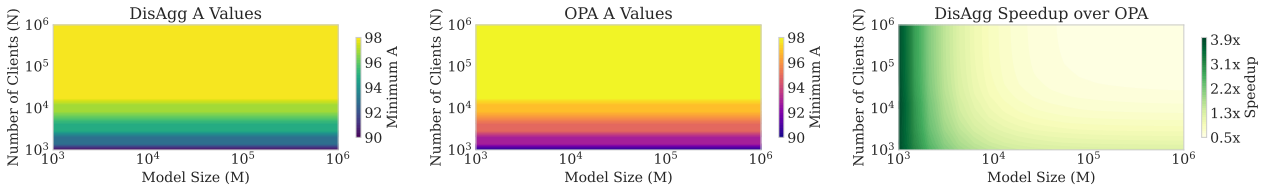


Figure 2. Contour plot showing the minimum number of Aggregators A for DISAGG and OPA and the resulting speedup of DISAGG over OPA across (M, N) under a 5G client connectivity assumption (2 MBps upload / 20 MBps download), with $k = 0.3$ and $k_{\text{comp}} = 0.66$. The number of Aggregators are computed for packing factor $\rho = 16$, matching OPA's configuration.

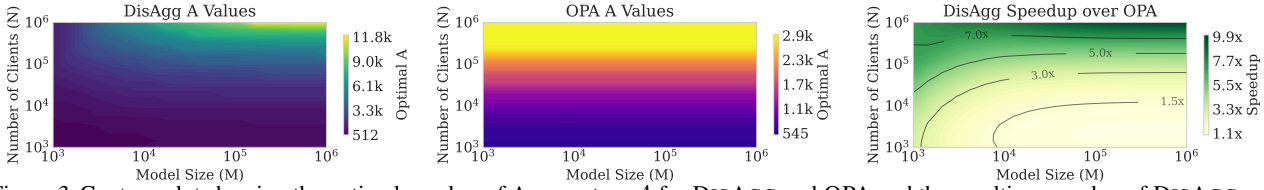


Figure 3. Contour plot showing the optimal number of Aggregators A for DISAGG and OPA and the resulting speedup of DISAGG over OPA across (M, N) under a 4G client connectivity assumption (200 kBps upload / 2 MBps download), with $k = 0.3$ and $k_{\text{comp}} = 0.66$.

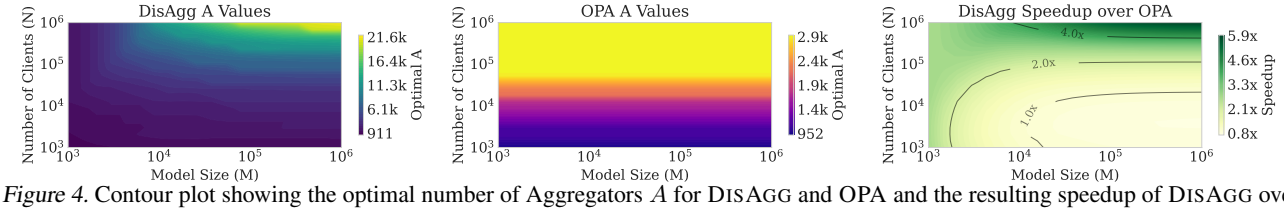


Figure 4. Contour plot showing the optimal number of Aggregators A for DISAGG and OPA and the resulting speedup of DISAGG over OPA across (M, N) under a 3G client connectivity assumption (50 kBps upload / 500 kBps download), with $k = 0.3$ and $k_{\text{comp}} = 0.66$.

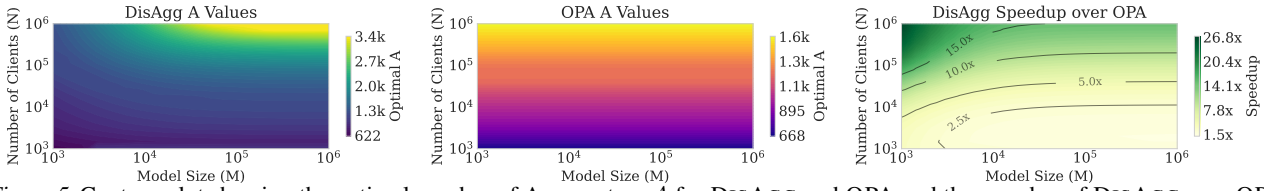


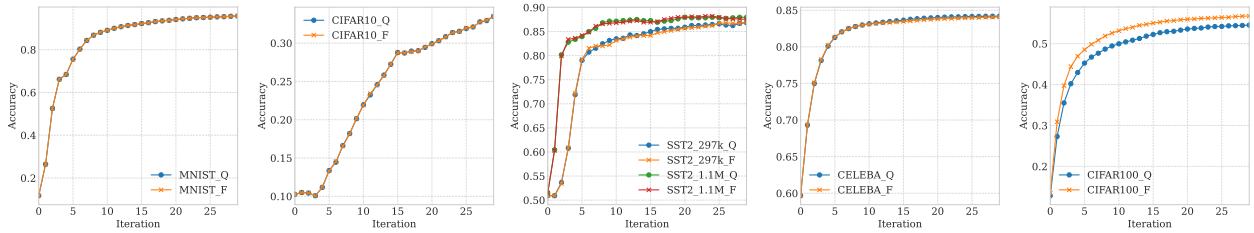
Figure 5. Contour plot showing the optimal number of Aggregators A for DISAGG and OPA and the speedup of DISAGG over OPA across (M, N) under the BFT constraint ($\gamma + \delta \leq 1/3$) and a 5G client connectivity assumption, with $\gamma = \delta = 0.1$ and $k_{\text{comp}} = 0.66$.

Table 3. Per stage timings in seconds for DISAGG, OPA and PLAINTEXT (simple aggregation) during one FL iteration. A DistilBERT model with 297k trainable parameters is trained on the SST2 dataset. DISAGG has a marginal overhead of 3.1% over PLAINTEXT.

Method	Setup	Client Comm	Committee Comm	Client Comp	Committee Comp	Server Comp	Total w/out Setup	Total
OPA	123.59	1.43	0.02	163.76	1e-3	72.96	238.17	361.76
DISAGG	0.75	7.07	1.79	66.27	0.09	0.19	75.41	76.16
PLAINTEXT	-	0.64	-	73.13	-	0.11	73.89	73.89

Table 4. Per stage timings in seconds for DISAGG, OPA and PLAINTEXT (simple aggregation) during one FL iteration. An EfficientNet model with 266k parameters is trained on the CelebA dataset. DISAGG has a marginal overhead of 20% over PLAINTEXT.

Method	Setup	Client Comm	Committee Comm	Client Comp	Committee Comp	Server Comp	Total w/out Setup	Total
OPA	109.81	1.29	0.02	291.68	2e-3	65.58	358.57	468.38
DISAGG	0.76	6.30	1.61	184.38	0.08	0.17	192.54	193.30
PLAINTEXT	-	0.57	-	160.34	-	0.12	161.03	161.03



(a) MNIST (b) CIFAR10 (c) SST2 (d) CELEBA (e) CIFAR100

Figure 6. Training accuracy with different datasets and model using plaintext FL. Q denotes the use of quantization required for cryptographic primitives in secure aggregation. F denotes floating point precision as used in standard FL.

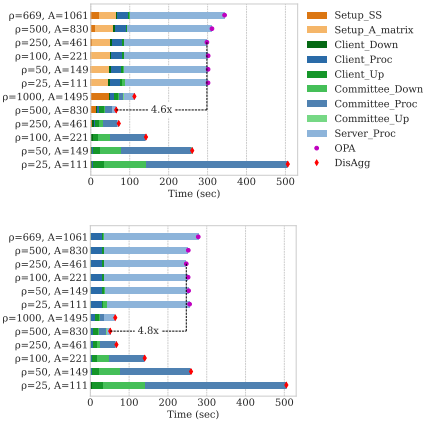


Figure 7. Speedup of DISAGG over OPA for one FL iteration with $M = N = 100k$ using a grid search of ρ as described in the main paper. A is calculated using a given value for ρ.

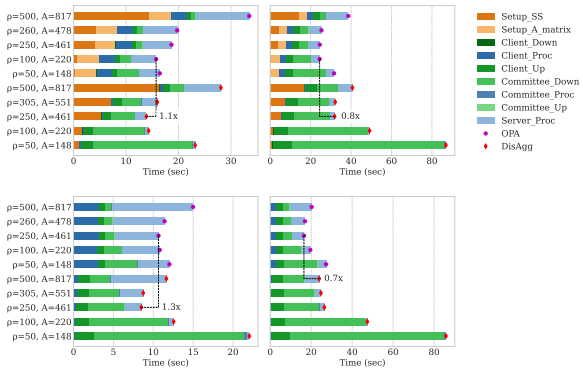


Figure 8. Time comparison for one FL iteration of DISAGG and OPA with (60% 5G, 40% 4G) clients on the left, (60% 5G, 40% 3G) on the right and a shared y-axis for each row. The top row includes the setup times while the bottom row excludes it. DisAgg is within $\pm 30\%$ of OPA timings under such conditions.

D.4 Effect of Heterogeneity

The main paper discusses the effects of stragglers on DISAGG's performance. In this section, we compare the effect of network heterogeneity on both DISAGG and OPA, by analyzing the effects of clients having a distribution of network speeds. Figure 8 presents cases with (60% 5G, 40% 4G) and (60% 5G, 40% 3G) client speed distributions. In the first case, DISAGG is still faster than OPA by 10-30%, whereas in the second case of having 40% clients with 3G connectivity, delays on the Aggregators' download of secret shares in DISAGG lead to a 20-30% slowdown compared to OPA. Given that 93% of the global population has access to at least 4G connectivity today (Amos, 2025), the second scenario represents an extremely unlikely case. In practical scenarios, the chances of encountering 3G clients may be no more than 10%; the server could either never select such clients initially or regard them as dropouts later. As shown in main paper, in such cases, DisAgg achieves a $1.37\times$ speedup over OPA after 30 iterations of FL training.

D.5 SecAgg+/LightSecAgg Offline Processing

LIGHTSECAGG (So et al., 2022) introduces offline computation phases for clients – allowing clients to compute mask shares in parallel to the FL training iteration. This effectively absorbs mask computation time into the total time spent for other phases. Figure 9 simulates the effects of using offline phases for SECAGG+ and LIGHTSECAGG by including or excluding mask computation time from the per iteration total time. As seen, parallelizing mask creation has a negligible impact on the total time as bottlenecks lie in other phases, which can be resolved by using DISAGG.

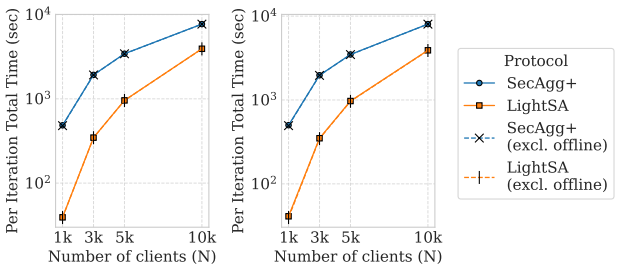


Figure 9. Combined computation and communication time per FL iteration for SECAGG+ and LIGHTSECAGG (LIGHTSA) for M=1k (left) and M=10k (right). Solid lines include mask computation time for SECAGG+ and LIGHTSECAGG while dashed lines exclude it. The difference in per iteration total time is negligible.

E ARTIFACT APPENDIX

E.1 Abstract

This artifact provides the code and accompanying instructions required to reproduce the experimental results presented in the paper. The code includes Python implementations of the baseline methods compared with DisAgg and scripts to run experiments on the various datasets and models evaluated. A `README.md` file describes how to install the required dependencies, prepare the datasets, and execute the experiments. The instructions allow users to replicate the evaluation procedure and regenerate the performance metrics reported in the paper, including accuracy and execution time, subject to variations due to hardware differences.

E.2 Artifact check-list (meta-information)

- **Algorithm:** DisAgg
- **Program:** Python 3.10 and various packages.
- **Model:** CNN, TinyNet, EfficientNet, DistilBERT+LoRA
- **Data set:** MNIST, CIFAR10, CIFAR100, CelebA, and SST2
- **Run-time environment:** Linux 64-bit with Nvidia drivers, Python 3.10 and Pip installed.
- **Hardware:** CPU with 10+ cores, GPU with at least 10 GB VRAM, at least 128 GB RAM, and 0.5 TB disk space
- **Metrics:** Accuracy, Time, Communication Size
- **Experiments:** see `README.md`
- **How much disk space required (approximately):** 0.5 TB
- **How much time is needed to prepare workflow (approximately):** 20mins
- **How much time is needed to complete experiments (approximately):** Few minutes up to many hours, depending on hardware and experiment.
- **Publicly available:** Yes
- **Code licenses:** CC-BY-NC 4.0
- **Data licenses:** See individual dataset URLs in `README.md`.
- **Workflow framework used:** None
- **Archived:** DOI on Zenodo

E.3 Description

E.3.1 How delivered

All instructions and code can be found using this publicly available URL: `https://github.com/SamsungLabs/mlsys26_disagg`. `README.md` inside the root directory contains instructions on how to install the required packages, prepare data and run experiments.

E.3.2 Data sets

The following datasets are used: MNIST, CIFAR-10, CIFAR-100, CelebA & SST-2. Instructions on how to download them can be found in the repo's `README.md`.

E.4 Installation

`README.md` contains instructions on how to install the required packages.

E.5 Experiment workflow

All experiments are configured in `src/constants.py`. To run a protocol with experiment index `<i>`:

```
python -m disagg_test --exp_index=<ix>
python -m opa_test --exp_index=<i>
python -m light_secagg_test --exp_index=<i>
python -m secagg_plus_test --exp_index=<i>
```

Each experiment index `<i>` corresponds to a specific result in a paper. All results are written to `outputs/`. Additional details on the experimental configuration and mapping between experiment indices and paper results can be found in the `README.md`.

E.6 Experiment customization

Experiments can be customized by modifying the configuration parameters in the source code. Additional details on how to change parameters and define new experimental settings are provided in the `README.md`.

E.7 Evaluation and expected result

`README.md` contains details on what experiments to run to replicate paper results. Note that timings would naturally vary based on hardware. However, we envision the comparative conclusions will be the same.

E.8 Methodology

Submission, reviewing and badging methodology:

- http://cTuning.org/ae/submission-20190109.html
- http://cTuning.org/ae/reviewing-20190109.html
- https://www.acm.org/publications/policies/artifact-review-badging

REFERENCES

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., et al. Deep learning with differential privacy. *CCS '16*, pp. 308–318, October 2016. URL https://doi.org/10.1145/2976749.2978318.

Amos, K. 4G network reaches 7.6b people as 5G penetration hits 54%, September 2025. URL https://guardian.ng/technology/4g-network-reaches-7-6b-people-as-5g-penetration-hits-54/.

Bell-Clark, J., Gascón, A., Li, B., Raykova, M., and Schoppmann, P. Willow: Secure aggregation with one-shot clients. Cryptology ePrint Archive, Paper 2024/936, 2024. URL https://eprint.iacr.org/2024/936.

Cooley, J. W. and Tukey, J. W. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965. URL https://doi.org/10.2307/2003354.

Geyer, R. C., Klein, T., and Nabi, M. Differentially private federated learning: A client level perspective. *arXiv preprint*, 2018. URL https://arxiv.org/abs/1712.07557.

Kadhe, S., Rajaraman, N., Koyluoglu, O. O., and Ramchandran, K. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint*, 2020. URL https://arxiv.org/abs/2009.11248.

Karthikeyan, H. and Polychroniadou, A. OPA: One-shot private aggregation with single client interaction and its applications to federated learning. *Cryptology ePrint Archive*, 2024. URL https://eprint.iacr.org/2024/723. Paper 2024/723.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. *arXiv preprint*, 2020. URL https://arxiv.org/abs/1907.02189.

Ma, Y., Woods, J., Angel, S., Polychroniadou, A., and Rabin, T. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. *Cryptology ePrint Archive*, 2023. URL https://eprint.iacr.org/2023/486. Paper 2023/486.

Mugunthan, V., de Carli Silva, B., and et al., P. Tacita: Secure aggregation for asynchronous federated learning with malicious servers. In *Proceedings of the 2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2025. URL https://ieeexplore.ieee.org/document/10710841.

Shamir, A. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. New York, NY, USA.

So, J., He, C., Yang, C.-S., Li, S., et al. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *arXiv preprint*, 2022. URL https://arxiv.org/abs/2109.14236.

Wei, K., Li, J., Ding, M., Ma, C., et al. Federated learning with differential privacy: Algorithms and performance analysis. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.00222.

Yu, H., Yang, S., and Zhu, S. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. *arXiv preprint*, 2018. URL https://arxiv.org/abs/1807.06629.

Yu, Q., Li, S., Raviv, N., et al. Lagrange coded computing: Optimal design for resiliency, security, and privacy. *AISTATS 2019*, pp. 1215–1225, 2019.

Zhang, X., Li, Z., Wan, K., Sun, H., Ji, M., and Caire, G. Fundamental limits of hierarchical secure aggregation with cyclic user association. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2503.04564.